04

Aplicando finally

Transcripción

[00:00] ¿Qué tal? Bienvenidos. Ya sabemos aquí que yo puedo abrir una conexión con try/catch y finally, yo puedo asegurarme pues que ejecute algún método, si ese método da algún error, catch lo va a atrapar, y finally pues va a ejecutar alguna otra cosa que yo quiera que ejecute, haya o no haya error. Finally es opcional. No es obligatorio tener la estructura de try/catch finally, pero sí es obligatorio un try/catch.

[00:38] Ahora, yo voy a hacer una pequeña modificación aquí para explicarles el siguiente concepto. Pues yo voy a instalar una conexión como nula y yo voy a crear aquí esta conexión. Con = new Conexión. Perfecto. Voy a arreglar el código aquí, listo, entonces la conexión aquí está nula. Yo voy a crear aquí la conexión.

[01:09] Y aquí yo voy a leer los datos, perfecto. Y aquí yo también podría hacer otra cosa. Yo puedo tener también un try solo con finally. También es permitido. Pregunta de examen de certificación. ¿Un try puede vivir solo? No, un try no puede vivir solo. ¿Un try puede vivir con un finally? Sí, compila. ¿Con catch? Con catch también, cuántos quieras.

[01:42] ¿Dos finales por ejemplo o algo así? No compila, solo hay un solo finally. ¿Finally y catch aquí abajo? No compila. Primero, debería ser el catch, entonces esa es la la primera cosa. Esa es la estructura más o menos, más o menos que tiene este bloque try/catch. Primero es try/catch, opcionalmente finally o try/finally. Vamos a volver aquí al tema. Si yo aquí lanzo la excepción

[02:25] Vamos a ejecutar aquí, guardamos. Vemos pues que abre la conexión, recibe los datos y cierra la conexión. No está dando el excepcion. ¿Qué pasaría si yo hago esto de aquí? Él cerró la conexión, pero dio un exception aquí. Curioso. En el finally él dio una excepción. ¿Y cuál es? IllegalStateException en conexión en este método. De todas formas, él pues reventó la bomba, y cerró la conexión.

[03:05] Llamó a su método cerrar y cerró la conexión, pero ¿qué sucedería si yo lanzo esta excepción aquí en el constructor? ¿Es válido? Sí. Es totalmente válido, estoy lanzando la excepción al momento de construir el objeto. Por lo tanto, ¿este objeto será creado o no será creado? Vamos a averiguarlo.

[03:37] Nuevamente, él había abierto conexión, recibió datos, ejecutó finally, cerró la conexión y de todas formas pues explotó la bomba. ¿Por qué? Por el finally. Como no teníamos un catch para atrapar y desactivar la bomba, solo tenemos un finally, entonces de todas formas explotó pero finally se ejecutó, cerró la conexión. Si yo vuelvo a lanzar este código vamos a encontrar otro error.

[04:04] ¿Y cuál es? Abre la conexión, perfecto y ejecuta finally directamente. ¿Por qué? Porque él dio un NullPointerException y vemos que la conexión nunca cerró. ¿Por qué nunca cerró? Porque ni siquiera se ha creado. Recordemos que la conexión aquí está null.

[04:24] Como la conexión aquí es nula y yo la estoy creando aquí, yo necesitaría que la conexión ya esté instanciada, ya esté creada, para llamar al método cerrar, si no este método de aquí va a llamar a un NullPointerException. Hasta ahí estamos todos en la misma página. ¿Entonces yo cómo conseguiría aquí tratar este problema, por ejemplo? Yo voy a regresar aquí el catch, primero, para atrapar la excepción que estoy lanzando aquí.

[05:16] Entonces yo ya me voy a asegurar de que voy a atrapar IllegalStateException cuando él esté creando la conexión. Perfecto. Entonces

nuevamente ejecutó y él ejecutó el finally, perfecto. Agarró IllegalStateException, imprimió, pero también el NullPointerException y nuevamente no está cerrando la conexión. ¿Por qué? Porque es nula. Recordemos que ahora la la bomba está explotando en el momento de la creación del objeto.

[05:48] ¿Cómo puedo yo asegurarme de cerrar la conexión? Se me ocurre algo como esto. Si conexión es diferente de null, entonces llama a cerrar. ¿Qué les parece? Entonces, con esto, ya me aseguro de que no voy a tener un NullPointerException o por lo menos esa debería ser la idea. Excelente. Abrí la conexión, explotó la bomba, la atrapó el catch, imprimo el StackTrace y como la conexión es nula, entonces él no llama al método cerrar.

[06:33] Podría ser que aquí yo le ponga un else con igual new Conexión y aquí hacer un con.cerrar. Podría ser, pero aquí yo estoy ya abriendo una nueva conexión y estoy cerrando una nueva conexión que no tiene nada que ver con esto. Entonces esta opción no es válida por ahora. Y si aquí yo, nuevamente, vamos a testear, vamos a probar ahora el otro caso, nuestro caso de éxito. Cierro aquí. Guardo aquí.

[07:12] Ejecuto. Perfecto. Perfecto. ¿Por qué? Porque creé la conexión, no hubo error, leí los datos, no hubo error. No hay catch, no se acciona el catch porque no hay error. Y finalmente, la conexión es nula. No, no es nula. Entonces, ahora sí puedes cerrar la conexión. Pero nuevamente aquí ya vemos aquí una validación dentro del finally.

[07:38] El código ya está comenzando a entreverarse un poco. Estructuras como esta, donde creamos el objeto con null aquí encima son muy comunes. ¿Por qué?

[07:50] Porque en el código, generalmente cuando cuando va evolucionando y lo toca un programador, otro programador y otro más, cada uno va adicionando su lógica y generalmente vamos a encontrarnos con situaciones

en las que el try inicializa un nuevo objeto, ya sea porque en la inicialización puede haber un tipo de exception que pueda ser lanzado, como fue este caso que hemos provocado aquí.

[08:19] Y eso podría conllevar también pues a tener ifs dentro del finally, ifs dentro del try, ifs dentro del catch también puede ser. Y ahí el código ya comienza a volverse un poco entreverado, lo que llamamos pues código espagueti, código fuertemente acoplado donde solo yo sé qué he hecho aquí, y si llega otro programador y quiere leer mi código sin que yo se lo explique, quizás no lo entienda, quizás no sepa por qué hice esto.

[08:50] ¿Entonces existe una forma de optimizar aún más este código existe? Existe. Exiuste, y se las voy a enseñar en el próximo video. Nos vemos.